

# Memory vectors for similarity search in high-dimensional spaces

Ahmet Iscen, *Member, IEEE*, Teddy Furon, *Member, IEEE*, Vincent Gripon, *Member, IEEE*,  
Michael Rabbat, *Member, IEEE*, and Hervé Jégou, *Member, IEEE*

**Abstract**—We study an indexing architecture to store and search in a database of high-dimensional vectors. This architecture is composed of several memory units, each of which summarizes a fraction of the database by a single representative vector. The potential similarity of the query to one of the vectors stored in the memory unit is gauged by a simple correlation with the memory unit’s representative vector. This representative optimizes the test of the following hypothesis: the query is independent from any vector in the memory unit vs. the query is a simple perturbation of one of the stored vectors.

Compared to exhaustive search, our approach finds the most similar database vectors significantly faster without a noticeable reduction in search quality. Interestingly, the reduction of complexity is provably better in high-dimensional spaces. We empirically demonstrate its practical interest in a large-scale image search scenario with off-the-shelf state-of-the-art descriptors.

**Index Terms**—High-dimensional indexing, image indexing, image retrieval.

## I. INTRODUCTION

WE consider the problem of searching for vectors similar to a query vector in a large database. In this context, many papers report how the curse of dimensionality (due to the size of the vectors) makes indexing techniques ineffective [1], [2]. The recent paper [2] describing and analyzing the popular FLANN method experimentally observes that even this state-of-the-art method performs poorly on synthetic high-dimensional vectors, and the authors conclude that “random datasets are one of the most difficult problems for nearest neighbor search.”

Some strategies have been proposed to (partly) overcome this problem. For instance, the vector approximation file [1] first relies on exhaustive search with approximate measurements and then computes the exact similarities only for a subset of vectors deemed of interest. The cosine sketch [3] approximates cosine similarity with faster Hamming distance. Other works like spectral hashing [4], Euclidean sketches [5], product quantization [6] and inverted multi-index [7] also rely on compact codes to speed up neighbor search while compressing the data. An interesting strategy is the Set Compression Tree [8], which uses a structure similar to a k-d tree to compress a set of vectors into an extremely compact representation. Instead of trying to explicitly return the vector,

the algorithm uses the structure to determine if a vector similar to a given query has been stored. Again, this method is dedicated to small dimensional vectors (its authors recommend the dimension be smaller than  $\log_2(N)$  where  $N$  is the size of the database) so that it is used in conjunction with a drastic dimension reduction via PCA to work with classical computer vision descriptors.

This paper proposes a similarity search approach specifically adapted to high-dimensional vectors such as those recently introduced in computer vision to represent images [9], [10]. The proposed indexing architecture consists of memory units, each of which is associated with several database vectors. A representative, called a memory vector, is produced for each memory unit and defined such that one can quickly and reliably determine whether or not at least one similar vector is stored in this unit by performing a single inner product with a new query.

Our problem is similar to the descriptor pooling problem in computer vision, but at a higher level. Many successful descriptors, such as BOV [11], [12], VLAD [10], FV [9], and EMK [13], encode and aggregate a set of local features into global representations. Yet, the new representation has a larger dimension than the local features. We solve a similar problem at a higher-level: instead of aggregating local features into a global image representation, we aggregate global representations into group representations but keeping the same ambient dimension and for the purpose of performing efficient search.

This paper studies similarity search from the perspective of statistical signal processing and decision theory while most of the experimental work uses computer vision large datasets. Its organization is as follows. Section II focuses on the design of a single memory vector. We formalize the similarity of a query with the vectors of one memory unit as a hypothesis test. We derive the optimal representative vector under some design constraints and show how to compute it in an online manner. Section III proposes and analyzes two different ways to assign database vectors to memory units: random assignment and weakly supervised assignment which packs similar vectors into memory units. We provide a theoretical and experimental analysis of the different design and assignment strategies. Finally, the experimental section IV evaluates our approach on standard benchmarks for image search. We use descriptors (vectors describing images) extracted with the most recent state-of-the-art algorithm in computer vision [14]. Our results show the potential of our approach for this application.

A. Iscen and T. Furon are with Inria Rennes.  
V. Gripon is with Télécom Bretagne.  
M. Rabbat is with McGill University.  
H. Jégou is with Facebook AI Research.

## II. MEMORY VECTORS

A memory unit  $\mathcal{X}$  is defined as a set of  $n$  vectors  $\{\mathbf{x}_1, \dots, \mathbf{x}_n\}$  of dimension  $d$ . Our objective is to produce a representative, a so-called memory vector, such that, given a query vector  $\mathbf{Y}$  regarded as a random variable, we can efficiently perform a “similarity” test answering: *is  $\mathbf{Y}$  a quasi-copy of, or similar to, at least one of the vectors of the memory unit?* For the sake of analysis, this section assumes that all vectors follow a uniform distribution on the  $d$ -dimensional unit hypersphere. The similarity between two vectors  $\mathbf{x}$  and  $\mathbf{y}$  is defined as the inner product  $\mathbf{x}^\top \mathbf{y}$ . We model the query as a random vector  $\mathbf{Y}$  distributed according to one of the two laws:

- Hypothesis  $\mathcal{H}_0$ :  $\mathbf{Y}$  is not related to any vector in  $\mathcal{X}$ .  $\mathbf{Y}$  is then uniformly distributed on the unit hypersphere.
- Hypothesis  $\mathcal{H}_1$ :  $\mathbf{Y}$  is related to one vector in  $\mathcal{X}$ , say  $\mathbf{x}_1$  without loss of generality. We write this relationship as  $\mathbf{Y} = \alpha \mathbf{x}_1 + \beta \mathbf{Z}$ , where  $\mathbf{Z}$  is a random vector orthogonal to  $\mathbf{x}_1$  and  $\|\mathbf{Z}\| = 1$ . This means that  $\mathbf{Y}$  is more similar to  $\mathbf{x}_1$  as  $\alpha$  gets closer to 1. We have  $\alpha^2 + \beta^2 = 1$  because  $\|\mathbf{Y}\| = 1$ .

We look for a representation scheme satisfying the following design constraints. First, the set of vectors  $\mathcal{X}$  is summarized by a single vector of the same dimension,  $\mathbf{m}(\mathcal{X}) \in \mathbb{R}^d$ , called the *memory vector* and denoted by  $\mathbf{m}$  when not ambiguous. Second, the potential membership of query  $\mathbf{Y}$  to  $\mathcal{X}$  is tested by thresholding the inner product  $\mathbf{m}^\top \mathbf{Y}$ .

### A. Sum-memory vector: analysis

A very simple way to define the memory vector is

$$\mathbf{m}(\mathcal{X}) = \sum_{\mathbf{x} \in \mathcal{X}} \mathbf{x}, \quad (1)$$

where we assume that  $\mathcal{X}$  is composed of  $n$  different vectors. Albeit naive, this strategy offers some insights when considering high-dimensional spaces.

Appendix A derives the pdf of the score  $\mathbf{m}^\top \mathbf{Y}$  under  $\mathcal{H}_0$  when  $\mathbf{m}$  is a known vector. This score has expectation 0 and variance  $\|\mathbf{m}\|^2/d$ , and it is asymptotically distributed as  $\mathcal{N}(0, \|\mathbf{m}\|^2/d)$  as  $d \rightarrow \infty$ . This gives an approximate pdf of  $\mathbf{m}^\top \mathbf{Y}$  under  $\mathcal{H}_0$ . In contrast, under  $\mathcal{H}_1$ , the inner product equals

$$\mathbf{m}^\top \mathbf{Y} = \alpha + \alpha \mathbf{m}(\mathcal{X}')^\top \mathbf{x}_1 + \beta \mathbf{m}(\mathcal{X}')^\top \mathbf{Z}, \quad (2)$$

with  $\mathcal{X}' = \mathcal{X} - \{\mathbf{x}_1\}$ . This shows two sources of randomness: the interference of  $\mathbf{x}_1$  with the other vectors in  $\mathcal{X}$  and with the noise vector  $\mathbf{Z}$ . Assuming that  $\mathbf{Y}$  is statistically independent of the vectors in  $\mathcal{X}'$  (this implies that the vectors of  $\mathcal{X}$  are mutually independent), we have

$$\begin{aligned} \mathbb{E}_{\mathbf{Y}}[\mathbf{m}^\top \mathbf{Y} | \mathcal{H}_1] &= \alpha, \\ \mathbb{V}[\mathbf{m}^\top \mathbf{Y} | \mathcal{H}_1] &= \|\mathbf{m}(\mathcal{X}')\|^2/d. \end{aligned} \quad (3)$$

Assuming that  $\mathcal{X}$  is composed of  $n < d$  statistically independent vectors on the unit hypersphere also gives  $\mathbb{E}_{\mathcal{X}}[\|\mathbf{m}(\mathcal{X})\|^2] = n$  and  $\mathbb{E}_{\mathcal{X}'}[\|\mathbf{m}(\mathcal{X}')\|^2] = n - 1$ . To summarize, for large  $d$ , we expect the following distributions:

$$\mathcal{H}_0 : \quad \mathbf{m}^\top \mathbf{Y} \sim \mathcal{N}(0, n/d), \quad (4)$$

$$\mathcal{H}_1 : \quad \mathbf{m}^\top \mathbf{Y} \sim \mathcal{N}(\alpha, (n-1)/d). \quad (5)$$

Making a hard decision by comparing the inner product to a threshold  $\tau$ , the error probabilities (false positive and false negative rates) are given by

$$\mathbb{P}_{\text{fp}} \approx 1 - \Phi\left(\tau \sqrt{d/n}\right) \quad (6)$$

$$\mathbb{P}_{\text{fn}} \approx \Phi\left((\tau - \alpha) \sqrt{d/(n-1)}\right), \quad (7)$$

where  $\Phi(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^x e^{-t^2/2} dt$ .

The number of elements one can store in a sum-memory vector is linear with the dimension of the space when vectors are drawn uniformly on the unit hypersphere. This construction is therefore useful for high-dimensional vectors only, as opposed to traditional indexing techniques that work best in low-dimensional spaces.

If the vectors were pair-wise orthogonal, the dominant source of randomness (the interference between  $\mathbf{x}_1$  and vectors of  $\mathcal{X}'$ ) is cancelled in (2). The variance under  $\mathcal{H}_1$  reduces to  $\beta^2(n-1)/d$ . This prevents any false negative if  $\beta \rightarrow 0$ . We further exploit this intuition that orthogonality helps in the next section.

### B. Optimization of the hypothesis test per unit

We next consider optimizing the construction of the memory vector of a given set  $\mathcal{X}$ . Denote the  $d \times n$  matrix  $\mathbf{X} = [\mathbf{x}_1, \dots, \mathbf{x}_n]$ . We impose that, for all  $i$ ,  $\mathbf{x}_i^\top \mathbf{m}(\mathcal{X}) = 1$  exactly and not only in expectation, as assumed above. In other words,  $\mathbf{X}^\top \mathbf{m} = \mathbf{1}_n$  where  $\mathbf{1}_n$  is the length- $n$  vector with all entries equal to 1. Achieving this, when  $\mathbf{Y} = \mathbf{x}_1$ , we eliminate the interference with the remaining vectors in  $\mathcal{X}'$  which was previously the dominant source of noise. In other words, under  $\mathcal{H}_1$ , Eq. (2) becomes

$$\mathbf{m}^\top \mathbf{Y} = \alpha + \beta \mathbf{m}^\top \mathbf{Z}. \quad (8)$$

Under  $\mathcal{H}_0$ , the variance of the score remains  $\|\mathbf{m}\|^2/d$ . Therefore, the norm of the memory vector is the key quantity determining the false positive probability.

We thus seek the representation  $\mathbf{m}$  minimizing the energy  $\|\mathbf{m}\|^2$  subject to the constraint that  $\mathbf{X}^\top \mathbf{m} = \mathbf{1}_n$ . If multiple solutions exist, the minimal norm solution is given by the *Moore-Penrose pseudo-inverse* [15]:

$$\mathbf{m}^* = (\mathbf{X}^\top)^+ \mathbf{1}_n. \quad (9)$$

Since  $n < d$ ,  $\mathbf{m}^* = \mathbf{X}(\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{1}_n$ . If no solution exists,  $\mathbf{m}^*$  is a minimizer of  $\|\mathbf{X}^\top \mathbf{m} - \mathbf{1}_n\|$ . This formulation amounts to treating the representation of a memory unit as a linear regression problem [16] with the objective of minimizing over  $\mathbf{m}$  the quantity  $\|\mathbf{X}^\top \mathbf{m} - \mathbf{1}_n\|^2$ . Taking the gradient, setting it equal to zero, and solving for  $\mathbf{m}$  gives back  $\mathbf{m}^*$ . When possible, and for large  $d$ , this new construction leads to the distributions:

$$\mathcal{H}_0 : \quad \mathbf{m}^{*\top} \mathbf{Y} \sim \mathcal{N}(0, \|\mathbf{m}^*\|^2/d) \quad (10)$$

$$\mathcal{H}_1 : \quad \mathbf{m}^{*\top} \mathbf{Y} \sim \mathcal{N}(\alpha, \beta^2 \|\mathbf{m}^*\|^2/d). \quad (11)$$

The major improvement comes from the reduction of the variance under  $\mathcal{H}_1$  for small values of  $\beta^2$ , i.e.,  $\alpha \lesssim 1$ . Appendix B shows that if the vectors of  $\mathcal{X}$  are uniformly

distributed then  $\|\mathbf{m}^*\|^2$  is larger in expectation than the square norm of the naive sum representation from Section II-A. The reduction of the variance under  $\mathcal{H}_1$  comes at the price of an increase of the variance under  $\mathcal{H}_0$ . However, this increase is small if  $n/d$  remains small. For large  $d$ , we have

$$\mathbb{P}_{\text{fp}} \approx 1 - \Phi\left(\tau\sqrt{\frac{d}{n} - 1}\right), \quad (12)$$

$$\mathbb{P}_{\text{fn}} \approx \Phi\left(\frac{\tau - \alpha}{\beta}\sqrt{\frac{d}{n} - 1}\right). \quad (13)$$

Note that  $\beta = \sqrt{1 - \alpha^2}$  is a decreasing function of  $\alpha$ . Therefore, if  $\tau < \alpha$ ,  $\mathbb{P}_{\text{fn}}$  is a decreasing function of  $\alpha$ . In particular  $\mathbb{P}_{\text{fn}} \rightarrow 0$  when  $\alpha \rightarrow 1$  as claimed above. In contrast to the naive sum approach from Section II-A, there is no longer false negative when the query  $\mathbf{Y}$  is exactly one of the vectors in  $\mathcal{X}$ . This holds for any value of  $\tau < 1$  when  $\alpha = 1$ , so that the false positive rate can be as low as  $1 - \Phi(\sqrt{d/n - 1})$ .

**Remark.** This solution is identical (up to a regularization) to the “generalized max-pooling” method introduced to aggregate local image descriptors [17]. However in our case the aggregation is performed on the database side only. Our solution is moreover theoretically grounded by a hypothesis test interpretation.

### C. Weakly supervised assignment

We now analyze a scenario where the vectors packed in the same memory unit are random but no longer uniformly distributed over the hypersphere: There is some correlation among them. The vectors in a memory unit are now uniformly distributed over a spherical cap (see Appendices D and E). This models the effect of a pre-processing which analyses the database vectors in order to assign similar vectors to memory units. For instance, Section III-B uses the  $k$ -means algorithm to process batches of database vectors.

We derive the same analysis as in the previous subsection with expectations and variances which now depend on the angle of the spherical cap. These expressions are complex and their derivation is detailed in the appendices mentioned above. In summary, the Kullback-Leibler distance between the distributions of  $\mathbf{m}^\top \mathbf{Y}$  under both hypotheses increases as the spherical cap gets narrow. In other words, identifying the positive memory units becomes easier when we assign correlated vectors to the same memory unit. Interestingly, this mechanism helps the *sum* construction more than *pinv*, so that when the vectors are very correlated, both constructions indeed perform equivalently.

## III. EXPERIMENTAL INVESTIGATION

We next consider application scenarios where we need to store a large number  $N$  of vectors and perform similarity search. One memory vector is not sufficient to achieve a reliable test. We therefore consider an architecture that consists of  $M$  memory vectors. The search strategy is as follows. A given query vector is compared with all the memory vectors. Then we compare the query with the vectors stored in the memory units associated with the high responses, i.e., those likely to contain a similar vector.

### A. Experimental Setup

Our experimental investigations are carried out using synthetic data (vectors uniformly distributed over the hypersphere) as well as real data, which are described in this section.

**Datasets.** We use the Inria Holidays [18], Oxford5k [19], and UKB [20] image datasets in our experiments. Additionally, we conduct a large scale experiment by adding images from the Flickr1M [19] dataset to the Holidays dataset.

**Descriptors.** We use the state-of-the-art triangular embedding descriptor [14], denoted by  $\phi_\Delta$ . We use the off-the-shelf reference implementation provided by the authors, which can be found online.<sup>1</sup> Each image is represented by a feature vector. The only difference is that we do not apply the “powerlaw normalization” to better illustrate the benefit of the *pinv* technique for the memory vector construction compared to the *sum* (when applying the powerlaw normalization, both designs perform equally well since the vectors are nearly orthogonal). Ultimately, we have  $d = 8064$  (or  $d = 1920$ ) dimensional feature vectors for each image, obtained by using a vocabulary of size 64 (resp. 16).

We also experiment using deep learning features provided by Babenko *et al.* [21]. As explained in their paper, the performance for the UKB dataset drops with adapted features trained on the Landmarks dataset. Therefore, we use the original neural codes trained on ILSVRC for the UKB dataset, and the adapted features for Holidays and Oxford5k. These features have  $d = 4096$  dimensions.

### B. Random assignment

We suppose that the  $N$  vectors in the database are *randomly* grouped into  $M$  units of  $n$  vectors:  $N = nM$ . We aim at finding the best value for  $n$ . When the query is related to the database (i.e., under  $\mathcal{H}_1$ ), we make the following assumption:  $\alpha_0 < \alpha < 1$ , and we fix the following requirement:  $\mathbb{P}_{\text{fn}} < \epsilon < 1/2$ . Since  $\mathbb{P}_{\text{fn}}$  is a decreasing function of  $\alpha$ , we need to ensure that  $\mathbb{P}_{\text{fn}}(\alpha_0) = \epsilon$ . This gives us the threshold  $\tau$ :

$$\tau = \mu_{\mathcal{H}_1} + \sigma_{\mathcal{H}_1} \Phi^{-1}(\epsilon), \quad (14)$$

with  $\mu_{\mathcal{H}_1}$  and  $\sigma_{\mathcal{H}_1}$  being the expectation and the standard deviation of  $\mathbf{m}_j^\top \mathbf{Y}$  under  $\mathcal{H}_1$ . Note that  $\Phi^{-1}(\epsilon) < 0$  because  $\epsilon < 1/2$  so that  $\tau < \alpha$ . The probability of false positive equals  $1 - \Phi(\tau/\sigma_{\mathcal{H}_0})$  which depends on  $n$ , denoted by  $\mathbb{P}_{\text{fp}}(n)$ . This is indeed an increasing function for both memory vector constructions. Now, we decide to minimize the expectation of the total computational cost  $C_{\mathcal{H}_0}$  when the query is not related. We need to compute one inner product  $\mathbf{m}_j^\top \mathbf{y}$  per unit, and then to compute  $n$  inner products  $\mathbf{x}_i^\top \mathbf{y}$  for the units giving a positive detection. In expectation, there are  $M \cdot \mathbb{P}_{\text{fp}}(n)$  such units, and so

$$C_{\mathcal{H}_0} = M + M \cdot \mathbb{P}_{\text{fp}}(n) \cdot n = N(n^{-1} + \mathbb{P}_{\text{fp}}(n)). \quad (15)$$

The total cost is the sum of a decreasing function ( $n^{-1}$ ) and an increasing function ( $\mathbb{P}_{\text{fp}}(n)$ ).

For the random assignment strategy, there is a tradeoff between having a few big units ( $n$  large) and many small

<sup>1</sup><http://www.tinyurl.com/democratic-kernel>

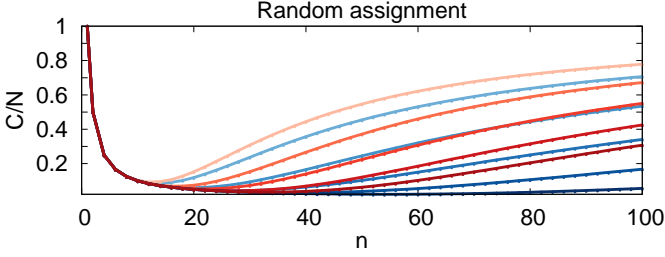


Fig. 1. Ratio of the global cost by the cost of the exhaustive search  $C_{H_0}/N$  as a function of  $n$  using random assignment on synthetic data. Setup:  $d = 1000$ ,  $\epsilon = 10^{-2}$ . The different curves correspond to values of  $\alpha_0 \in \{0.5, 0.6, 0.7, 0.8, 0.9\}$ . Red and blue lines correspond to *sum* and *pinv* constructions respectively. Darker shades correspond to higher  $\alpha_0$ .

units ( $n$  small). Fig. 1 illustrates this tradeoff for different values of  $\alpha_0$  with synthetic data. It is not possible to find a closed form expression for the cost minimizer  $n^*$ . When  $\alpha_0$  is close to 1, the threshold is set to a high value, producing reliable tests, and we can pack many vectors into each unit:  $n^*$  is large allowing a huge reduction in complexity. Even when  $\alpha_0$  is as small as 0.5,  $n^*$  is small but the improvement remains significant. In the setup of Fig. 1, the proposed approach has a complexity that is less than one tenth of that of searching through all database vectors (equivalent to  $n = 1$ ). However, in order to increase the efficiency, we introduce an additional  $\mathcal{O}(Md) = \mathcal{O}(dN/n)$  memory overhead for storing memory vectors.

Figure 2 depicts the theoretical and empirical Receiver Operating Characteristic (ROC) curves for different values of  $\alpha$ . For the synthetic data,  $\mathbb{P}_{fn}$  and  $\mathbb{P}_{fp}$  are evaluated using Eq. (6) and (12). As expected the test performs better when  $\alpha$  is closer to 1 and when  $n \ll d$ . For the real data, we use cosine similarity-based ground truth, since it is directly related with the model we considered theoretically. For each query vector, we deem a database vector as relevant if their cosine similarity is greater than  $\alpha_0$ . To have enough ground-truth vectors, we look for these matching vectors on the Holidays+Flickr1M dataset using various  $\alpha_0$  values. This experiment using real data confirms the findings of the theoretical analysis. The *pinv* construction performs better than the *sum* as long as  $\alpha_0$  is big as explained in Sect. II-B. However, the theoretical analysis is unable to predict performance levels on real dataset. It seems that the vectors of this real dataset have a much lower intrinsic dimensionality than their representational dimension  $d = 1920$ .

### C. Weakly supervised assignment

A well-known technique in the approximate search literature is to partition the space  $\mathbb{R}^d$  by clustering the database vectors. This assigns similar database vectors to the same cell [2]. In Section II-C, we explain the advantage of a weakly supervised assignment by showing that the distance between the distributions of positive and negative memory units similarities increases. To show this point experimentally, we modify the spherical  $k$ -means clustering [22], so that the clusters are

represented using *pinv* (or *sum*) in the update stage <sup>2</sup>.

**Better hypothesis test.** Figure 3 shows that highly ranked memory units are very likely true positives containing at least one matching vector. On the contrary, with the random assignment, a positive memory unit may have a low rank. This means that we now can analyse the database vectors of a shorter list of memory units to find most of the matching vectors.

**More than one match.** Another byproduct of weakly supervised assignment is that positive memory units are likely to contain more than one match, since matching vectors usually have high cosine similarity with each other. This helps the search efficiency by returning most of the matching vectors by only scanning a few positive memory units. This is also experimentally shown in Figure 4. With the random assignment, we have almost surely at most one matching vector in each positive memory unit.

**Imbalance factor.** We now analyze the cost of search with weakly supervised assignment. In Eq. (15), we assume that each unit contains  $n$  vectors. Up to this approximation, Fig. 5 shows that both constructions *sum* and *pinv* perform better as the inner correlation increases, but more surprisingly, they perform equivalently. It is also shown that it is possible to pack many vectors into the same memory unit with weakly supervised assignment, and still obtain a low search cost.

In practical applications, assuming that each unit contains a constant number of vectors is no longer true with weakly supervised assignment. This makes the analysis of the complexity more involved than (15). Moreover, this is potentially problematic in some applications: the complexity and thus the runtime can change dramatically from one query to another.

Imbalance factor is a metric to measure the impact of unbalanced clusters [23]. It is defined as

$$\delta = M \sum_{i=1}^M p_i^2, \quad (16)$$

where  $M$  is the number of clusters, and  $p_i$  is the empirical probability that a database vector belongs to the  $i$ -th cluster. This is measured as frequency  $p_i = n_i/N$ , where  $n_i$  is the cardinality of the  $i$ -th cluster. Simple derivations give the following expectation and variance:  $\mathbb{E}(n_i) = N/M$  and  $\mathbb{V}(n_i) = (\delta - 1)N^2/M^2$ . This shows that higher imbalance factor corresponds to clusters with varying sizes. This gives birth to a wide variability of the complexity from one query to another.

Table I shows the imbalance factor for different weakly supervised assignments. It is shown that *pinv* variants have more balanced cells compared to traditional *sum*, making the search process more effective. The negative effect of high imbalance factor in practice is better observed in Figure 6. In this figure, the algorithm visits a fixed number of positive memory units: 7 (Holidays), 30 (Oxford5k), or 60 (UKB). This roughly gives us a complexity ratio of  $C_{H_0}(\tau) \approx 0.2$

<sup>2</sup>Note that, when we use such assignment techniques in spherical  $k$ -means, the number of vectors per clusters is not evenly distributed. The dot product may be dominated by long cluster representation vectors. Hence, we also propose a normalized version of the assignment, where the cluster representation vectors obtained are normalized to the unit norm.

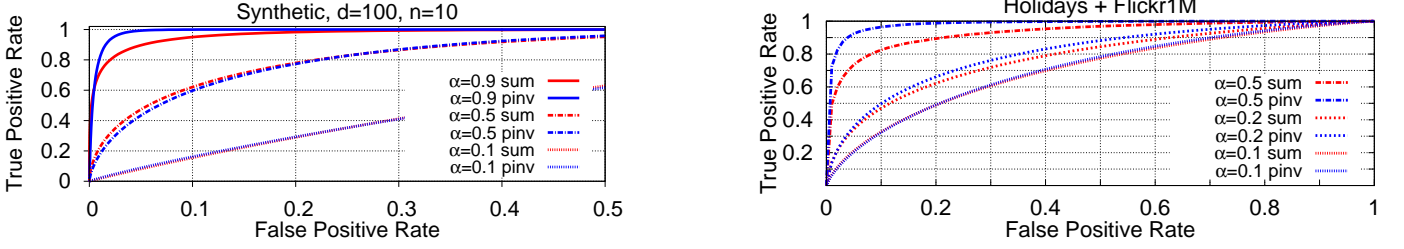


Fig. 2. ROC curves ( $1 - \mathbb{P}_{\text{fn}}$  as a function of  $\mathbb{P}_{\text{fp}}$ ) for *sum* and *pinv* constructions evaluated using (left) synthetic data with  $d = 100$  and  $n = 10$ , (right) Holidays+Flickr1M real dataset and  $\phi_{\Delta}$  features with  $d = 1920$  and  $n = 10$ .

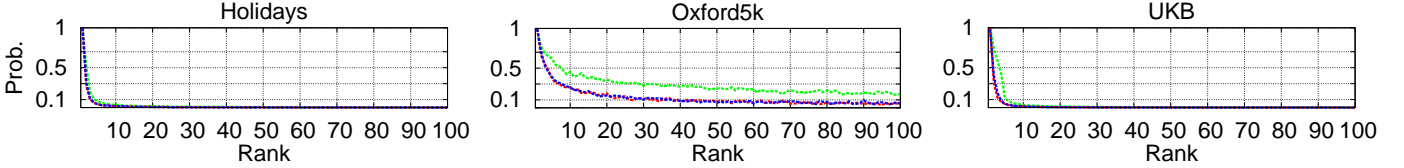


Fig. 3. Probability that a memory unit contains at least one match with respect to their rank. Green, red and blue lines correspond to random, *sum* spherical k-means and *pinv* spherical k-means respectively. There is a high probability of retrieving a match in highly ranked memory units, but it decreases faster with a weakly supervised assignment.

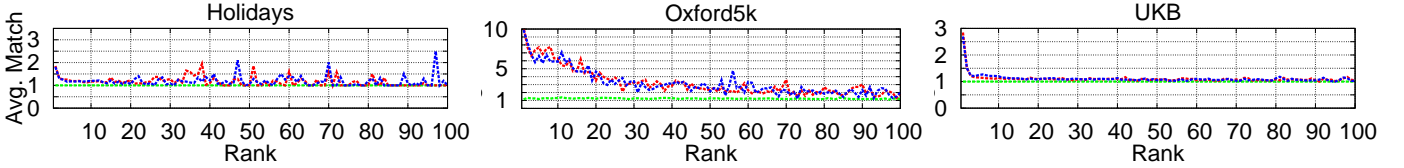


Fig. 4. The average number of matches given that the memory unit is positive. Green, red and blue lines correspond to random, *sum* spherical k-means and *pinv* spherical k-means respectively. Using random assignment, we have about 1 matching vector per memory unit. The weakly supervised assignment improves this especially for higher-ranked units.

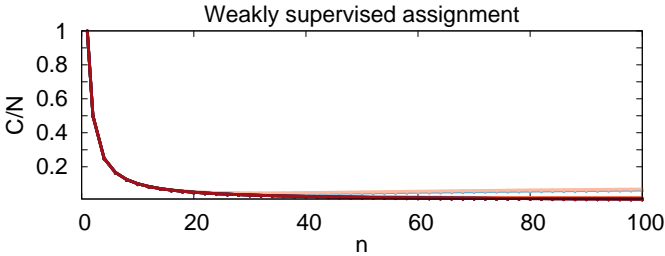


Fig. 5. Ratio of the global cost by the cost of the exhaustive search  $C_{\mathcal{H}_0}/N$  as a function of  $n$  using weakly supervised assignment and synthetic data. Setup:  $d = 1,000$ ,  $\epsilon = 10^{-2}$ . The different curves correspond to values of  $\alpha_0 \in \{0.5, 0.6, 0.7, 0.8, 0.9\}$ . Red and blue lines correspond to *sum* and *pinv* constructions respectively. Darker shades correspond to higher  $\alpha_0$ .

on average. We then show the complexity ratio per query in a histogram. It is clearly seen that the distribution for *pinv* has smaller standard deviation compared to *sum*, even though their means are almost the same. This makes *pinv* variant of spherical  $k$ -means a better alternative for weakly supervised assignment.

#### IV. APPLICATION TO IMAGE SEARCH

This section shows that memory vectors perform extremely well on typical computer vision benchmarks. We assume two scenarios: closed-datasets in Section IV-A, and large-scale and streaming data in Section IV-B.

	sum	sum + norm.	pinv	pinv + norm.
Holidays	2.08	2.17	<b>1.90</b>	2.03
Oxford5k	2.76	2.69	2.27	<b>2.23</b>
UKB	2.58	2.56	<b>2.06</b>	2.09

TABLE I

IMBALANCE FACTOR FOR DIFFERENT DATASETS USING *sum*, *pinv* AND THEIR NORMALIZED VARIANTS OF K-MEANS. EACH DATASET IS CLUSTERED INTO  $M = N/10$ .

Whereas the datasets were already introduced in Sect. III-A, let us describe the measure of performances. We follow the standard image retrieval protocol where each image is represented by a feature vector and the ground truth is now based on the visual similarity. The goal is to return visually similar images for a given query image. The similarity of two images is measured by the cosine of their descriptor vectors, and the images are ordered accordingly. We adopt the performance measure defined for each benchmark (mAP or 4-recall@4, the higher the better).

As for the complexity, we first measure the similarities between the query and  $M$  memory vectors. We compare these similarities with a given threshold  $\tau$ . Then, we re-rank all the vectors in positive memory units according to their similarities with the query vector. We characterize the complexity of the

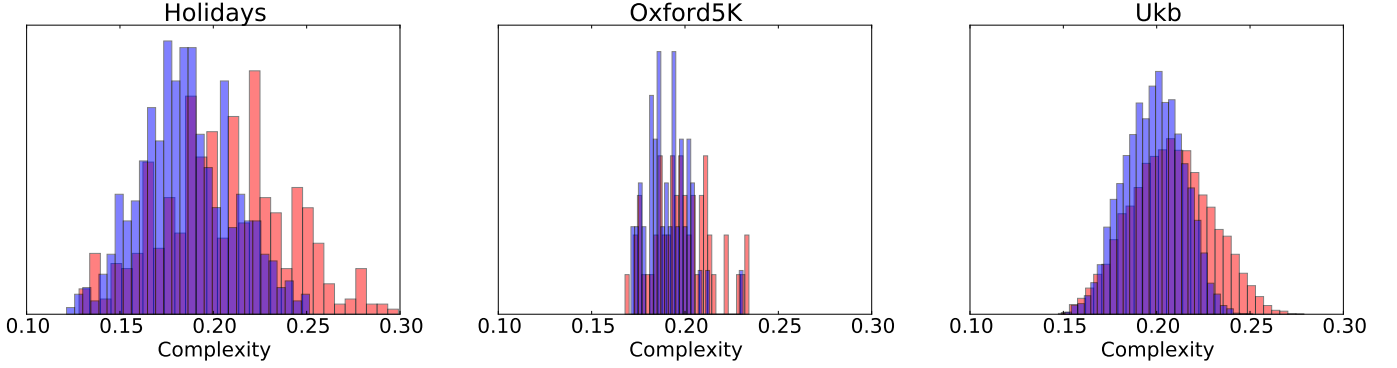


Fig. 6. Complexity per query for *pinv* k-means (blue) and *sum* k-means (red). Although the averaged complexity ratios over all queries are similar, *pinv* has less variance thanks to lower imbalance factor.

search per database vector by:

$$C_{\mathcal{H}_1}(\tau) = M + \sum_{i: \mathbf{y}^\top \mathbf{m}_i > \tau} n_i, \quad (17)$$

where  $n_i$  is the number of database vectors in the  $i$ -th memory unit. We measure the complexity ratio  $C_{\mathcal{H}_1}(\tau)/N$  and the retrieval performance for different values of the threshold  $\tau$ . For large  $\tau$ , no memory unit is positive, resulting in  $C_{\mathcal{H}_1}(\tau)/N = M$  and no candidate is returned. As  $\tau$  decreases, more memory units trigger reranking.

#### A. Closed dataset

Recall from Section III that weakly supervised assignment provides better approximate search than a random assignment. This is confirmed for the image search benchmark in Figure 7. Additionally, we show that it is possible to pack more vectors in a memory unit using weakly supervised assignment in Figure 8. We use this approach (spherical  $k$ -means with *pinv*) for the rest of our experiments.

**The dimensionality** of the descriptor linearly impacts the efficiency of any system. Dimensionality reduction with PCA is one way to improve this point. Our method is compatible with dimensionality reduction as shown in Fig. 9, where we reduce the vectors to  $d' = 1024$  components. The search performance is comparable to the baseline with less computational complexity. We also apply our method to features learned with deep learning ( $d = 4096$ ). Fig. 10 shows that the reduction in complexity also applies when using high performance deep learning features.

**Compact codes** are another way to increase efficiency. We reduce the dimensionality of the descriptor vectors to  $d' = 1024$  and binarize them by taking the *sign* of each component, in the spirit of cosine sketches [3]. In the asymmetric case [24], [25], only memory vectors and dataset vectors are binarized, whereas in the symmetric case query vectors are also binarized during the query time.

Figures 11 and 12 show the performance when using compact binary codes. For the symmetric case, the *sum* method seems to perform better than *pinv* on the Holidays and Oxford5k datasets. In the asymmetric case, both methods perform similarly. In all cases, we achieve convergence to the baseline with a complexity ratio well below 1. Implementation

efficiency is further improved in the symmetric case by using the Hamming distance calculation instead of dot product.

**Comparison with FLANN** [2]. Running the FLANN algorithm on the Holidays+Flickr1M dataset reveals that the convergence to the baseline is achieved with a speedup of 1.25, which translates to a complexity ratio of around 0.8. We can achieve similar performance with a complexity ratio of only 0.3. This confirms that FLANN is not effective for high-dimensional vectors. In this experiment, we use the autotune option of the FLANN library, and set `target_precision = 0.95`, `build_weight = 0.01`, and `memory_weight = 0`.

**Execution time.** We have shown that we get close to baseline performance while executing significantly fewer operations. We now measure the difference in execution time under a simple setup:  $d = 1024$  and  $N = 10^6$  dataset vectors. An average dot product calculation between the query and all dataset vectors is  $0.2728s$ . With  $N/10$  memory vectors and  $\approx 100k$  vectors in positive memory vectors, the execution time decreases to  $0.0544s$ . We improve the efficiency even further with symmetric compact codes and Hamming Distance computation: the execution time becomes  $0.0026s$ . Our method can be parallelized for even more improvement.

#### B. Large scale and streaming data

We conduct large scale experiments on Holidays+Flickr1M. The main advantage of our approach is its compatibility with large scale and streaming data, where pre-clustering the data may not be possible. More specifically, we assume that we have streaming images which we would like to index. As the size of the data keeps growing continuously, it is not possible to apply traditional  $k$ -means in such a scenario. We investigate two different approaches: random assignment and weakly supervised assignment over mini-batches.

**Online indexing** assumes that we would like to index items in streaming data as they become available. In such case, the random assignment is applicable provided that the successive vectors in the stream are independent.

Figure 13 shows the image retrieval performance based on random assignment with different group sizes  $n$ . With  $n = 10$ , the performance is close to the baseline while performing roughly three times fewer vector operations than exhaustive search. On the other hand, larger groups make it possible



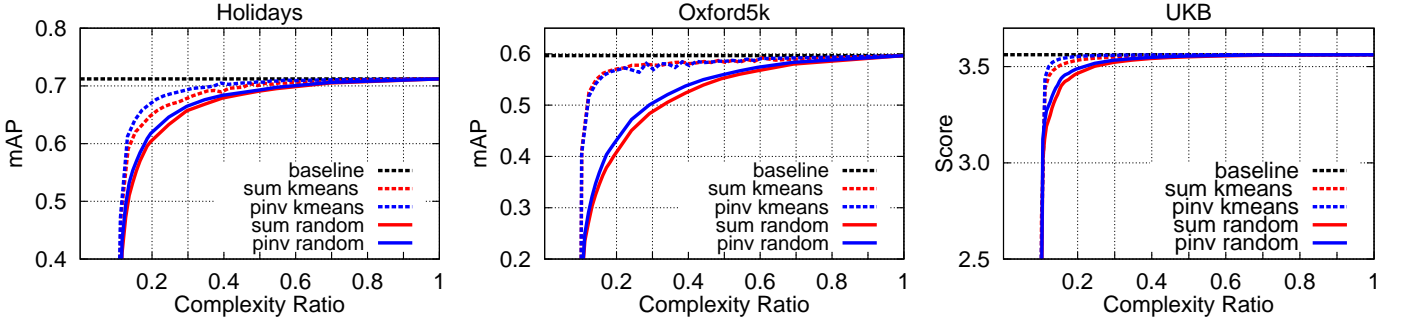


Fig. 7. Image retrieval performance using visual similarity ground truth. K-means variants bring significant improvement.

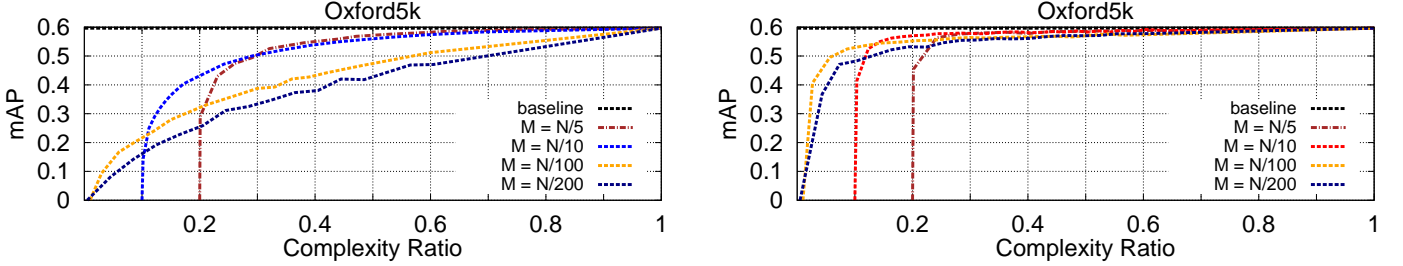


Fig. 8. Search performance using random assignment (left) and weakly supervised assignment (right). This last option uses fewer memory units and still obtains a good search performance, which is not possible with random assignment.

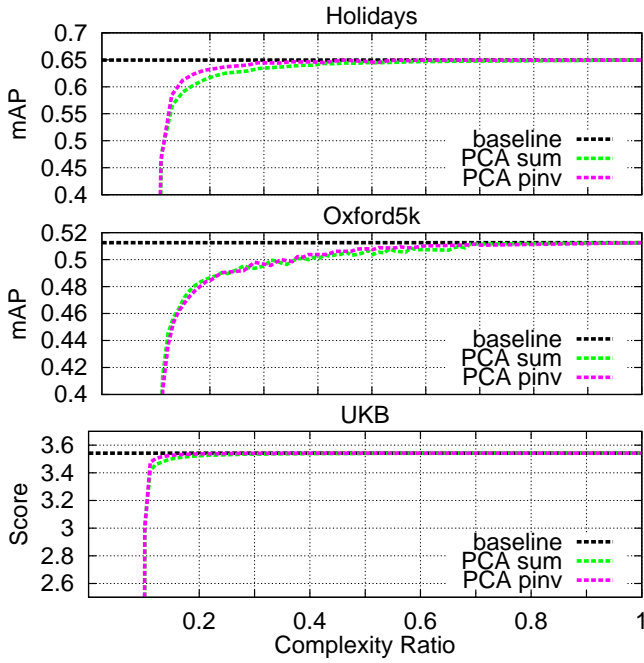


Fig. 9. The performance of memory vectors after PCA dimensionality reduction with  $d' = 1024$ .

to have smaller complexity ratio with a degrading effect on the quality of search, since the scores obtained from memory vectors are noisier (see (6) and (12)). The *pinv* construction performs better than *sum* in all cases except for a very large memory units of size  $n = 500$ , where the quality of search is low in general.

**Batch assignment.** The alternative approach runs weakly supervised assignment on small batches. A *batch* spherical *k*-

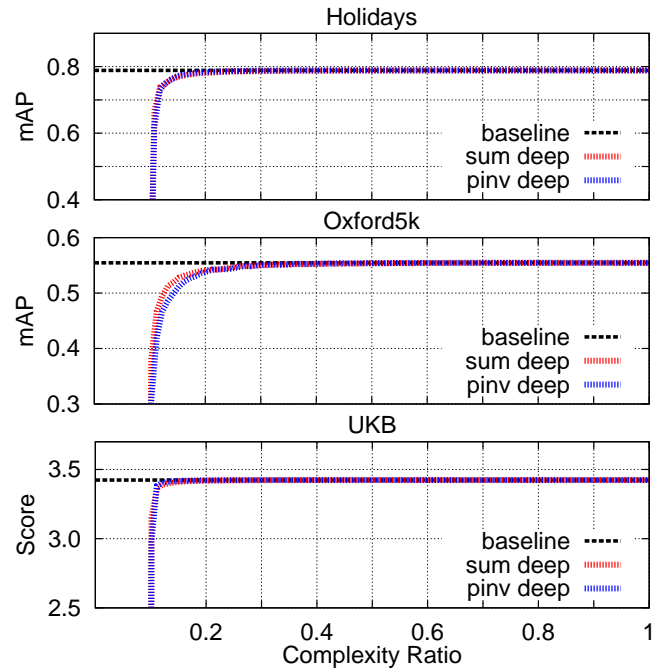


Fig. 10. Image retrieval performance with deep learning features ( $d = 4096$ ), as trained by Babenko *et al.* [21]. Features for Holidays and Oxford5K are retrained on the Landmarks dataset, whereas the ones for UKB are trained on ILSVRC.

means is the same as the regular spherical *k*-means discussed in previous sections. We do not cluster the whole dataset at once because this would not be tractable with large collections. Instead, we randomly divide the dataset into batches of the same size and run a weakly supervised assignment separately for each batch. Fig. 14 shows that this strategy improves the

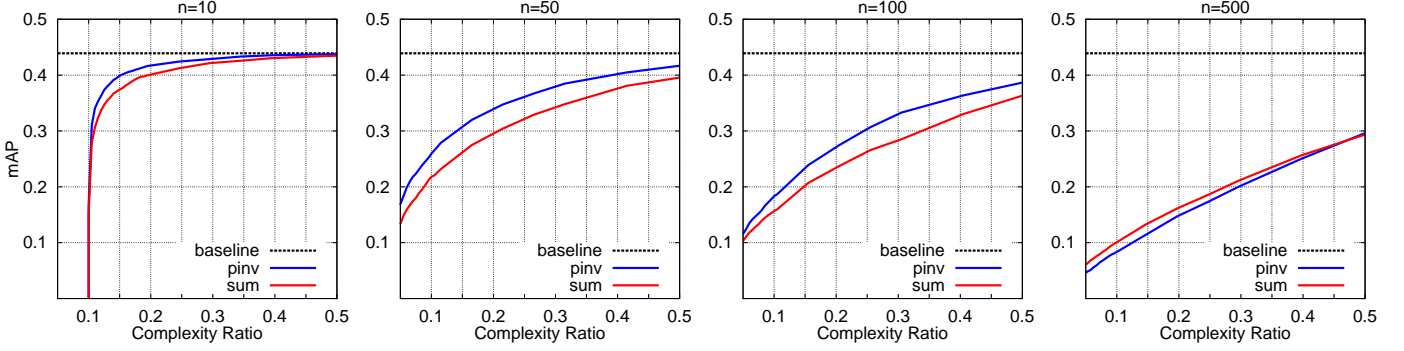


Fig. 13. Image retrieval performance in Holidays+Flickr1M with different memory unit size  $n$ . The data is streamed in this scenario, and the memory units are created randomly.

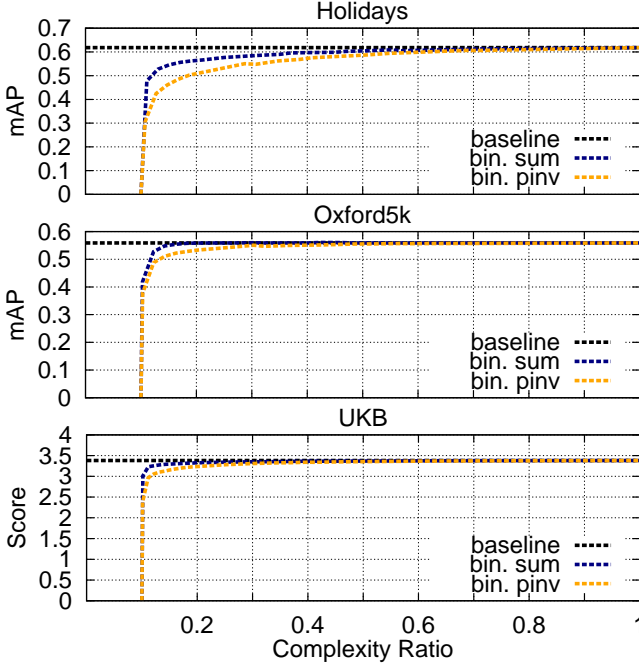


Fig. 11. Experiments with binary codes after PCA reduction to  $d' = 1024$ . The quantization is symmetric: real query vectors are binarized and then compared to binary memory vectors.

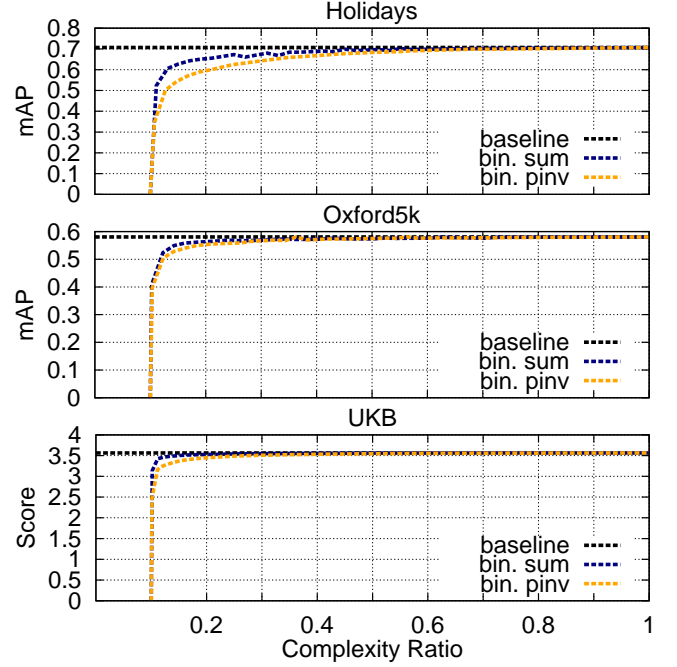


Fig. 12. Experiments with binary codes after PCA reduction to  $d' = 1024$ . The quantization is asymmetric: real query vectors are compared to binary memory vectors.

performance while keeping the complexity of the clustering algorithm manageable.

We compare our approach to a well-known mini-batch  $k$ -means algorithm [26] (referred to as mbk) as implemented by [27]. We compare both strategies using a batch size of 10k, and show the image retrieval performance for different complexity ratio in Figure 15.

The first observation is that the plot of mbk is not smooth. This is due to the clusters being unbalanced when using such mini-batch approaches. In fact, when we measure the imbalance factor (16), we obtain  $\delta = 183 \pm 8$ . As a result, few clusters contain a large number of dataset vectors, and when a cluster is accessed, the cost of the verification step becomes expensive. On the contrary, the imbalance factor observed using our batch spherical  $k$ -means is only  $2.47 \pm 0.01$ , resulting in a more efficient verification step for positive memory units.

The complexity ratio per query for the two methods can

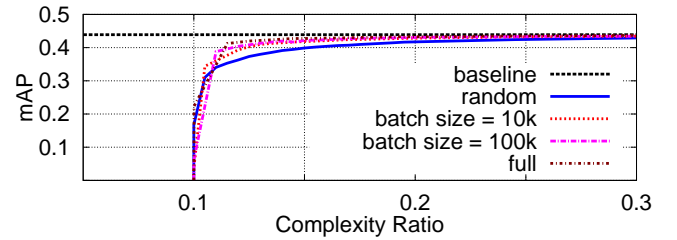


Fig. 14. Comparison of random and weakly supervised assignments over batches in a large-scale setup. Option *full* corresponds to a unique batch of size  $10^6$ . We run each experiment multiple times.

be seen in Figure 16. When we set the number of positive memory units to 3500, the mean complexity ratio for our batch spherical  $k$ -means approach is 0.17, with a standard deviation of 0.01. On the other hand, when mbk is used, the mean increases to 0.26 with a standard deviation of 0.51.



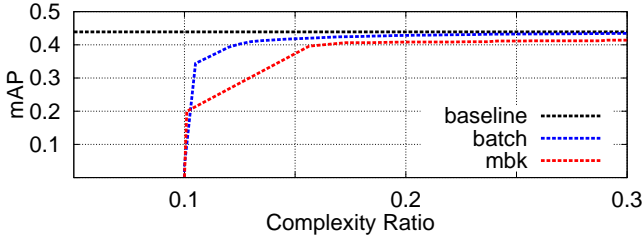


Fig. 15. Comparison of our *batch spherical k-means* approach with the mini-batch *k-means* algorithm (mbk) [26]. The batch size equals 10k.

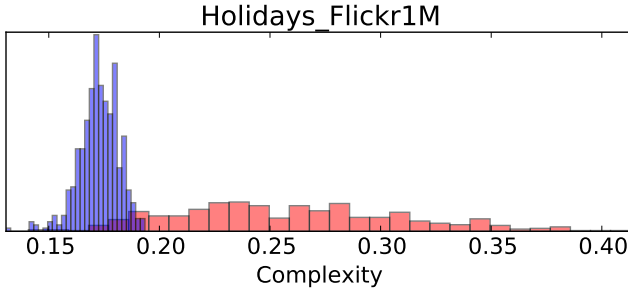


Fig. 16. Complexity of each query for Holidays+Flickr1M using our batch spherical *k-means* (blue) and mbk (red). Our scheme has less complexity on average and less variance.

## V. CONCLUSION

In this paper, we take a statistical signal processing point of view for image indexing, instead of traditional geometrical approaches in the literature. This shift of paradigm allows us to bring theoretical justifications for representing a set of vectors. We have presented and analyzed two strategies for designing memory vectors, enabling efficient membership tests for real-valued vectors. We have also showed two possible assignment strategies and analyzed their performance theoretically and experimentally. For random assignment, the optimized *pinv* construction gives better results than the simple *sum* aggregation. On the other hand, when the vectors in the same memory unit share some correlation, *sum* is on par with the *pinv* construction as for the quality of the hypothesis test. Yet, the *pinv* construction when used in the weakly supervised assignment offers a lower imbalance factor. This yields less variability of the search runtime from one query to another. This procedure is done offline and its complexity is often ignored in the image search literature. On the contrary, we did pay attention to this bottleneck: we proposed to run the weakly supervised assignment by batch and showed that it does not spoil the overall performance of the image search.

## REFERENCES

- [1] R. Weber, H.-J. Schek, and S. Blott, "A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces," in *Proceedings of the International Conference on Very Large DataBases*, 1998, pp. 194–205.
- [2] M. Muja and D. G. Lowe, "Scalable nearest neighbor algorithms for high dimensional data," *IEEE Trans. PAMI*, vol. 36, 2014.
- [3] M. S. Charikar, "Similarity estimation techniques from rounding algorithms," in *STOC*, May 2002, pp. 380–388.
- [4] Y. Weiss, A. Torralba, and R. Fergus, "Spectral hashing," in *NIPS*, December 2009.
- [5] W. Dong, M. Charikar, and K. Li, "Asymmetric distance estimation with sketches for similarity search in high-dimensional spaces," in *SIGIR*, July 2008, pp. 123–130.
- [6] H. Jégou, M. Douze, and C. Schmid, "Product quantization for nearest neighbor search," *IEEE Trans. PAMI*, vol. 33, no. 1, pp. 117–128, January 2011.
- [7] A. Babenko and V. Lempitsky, "The inverted multi-index," in *CVPR*, June 2012.
- [8] R. Arandjelović and A. Zisserman, "Extremely low bit-rate nearest neighbor search using a set compression tree," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2014.
- [9] F. Perronnin and C. R. Dance, "Fisher kernels on visual vocabularies for image categorization," in *CVPR*, June 2007.
- [10] H. Jégou, M. Douze, C. Schmid, and P. Pérez, "Aggregating local descriptors into a compact image representation," in *CVPR*, June 2010.
- [11] G. Csurka, C. Dance, L. Fan, J. Willamowski, and C. Bray, "Visual categorization with bags of keypoints," in *ECCV Workshop Statistical Learning in Computer Vision*, 2004.
- [12] J. Sivic and A. Zisserman, "Video Google: A text retrieval approach to object matching in videos," in *ICCV*, October 2003, pp. 1470–1477.
- [13] L. Bo and C. Sminchisescu, "Efficient match kernels between sets of features for visual recognition," in *NIPS*, 2009.
- [14] H. Jégou and A. Zisserman, "Triangulation embedding and democratic kernels for image search," in *CVPR*, June 2014.
- [15] C. R. Rao and S. K. Mitra, "Generalized inverse of a matrix and its applications," in *Proceedings of the Sixth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Theory of Statistics*, 1972.
- [16] C. M. Bishop, *Pattern recognition and machine learning*. Springer New York, 2006, vol. 1.
- [17] N. Murray and F. Perronnin, "Generalized max-pooling," in *CVPR*, June 2014.
- [18] H. Jégou, M. Douze, and C. Schmid, "Hamming embedding and weak geometric consistency for large scale image search," in *ECCV*, October 2008.
- [19] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman, "Object retrieval with large vocabularies and fast spatial matching," in *CVPR*, June 2007.
- [20] D. Nistér and H. Stewénus, "Scalable recognition with a vocabulary tree," in *CVPR*, June 2006, pp. 2161–2168.
- [21] A. Babenko, A. Slesarev, A. Chigorin, and V. Lempitsky, "Neural codes for image retrieval," in *ECCV*, 2014.
- [22] I. S. Dhillon and D. S. Modha, "Concept decompositions for large sparse text data using clustering," *Machine learning*, vol. 42, no. 1-2, pp. 143–175, 2001.
- [23] H. Jégou, M. Douze, and C. Schmid, "Improving bag-of-features for large scale image search," *IJCV*, vol. 87, no. 3, pp. 316–336, February 2010.
- [24] A. Gordo and F. Perronnin, "Asymmetric distances for binary embeddings," in *CVPR*, 2011.
- [25] M. Jain, H. Jégou, and P. Gros, "Asymmetric hamming embedding," in *ACM Multimedia*, October 2011.
- [26] D. Sculley, "Web-scale k-means clustering," in *Proceedings of the 19th international conference on World wide web*. ACM, 2010.
- [27] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg *et al.*, "Scikit-learn: Machine learning in python," *The Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.
- [28] A. R. DiDonato and M. P. Jarnagin, "A method for computing the incomplete beta function ratio," Computation and Analysis laboratory, U.S. Naval Weapons Laboratory, Dahlgren, VI, USA, Tech. Rep. No. 1949, Oct. 1966.
- [29] T. Groves and T. Rothenberg, "A note on the expected value of an inverse matrix," *Biometrika*, vol. 56, no. 3, pp. 690–691, Dec. 1969.

## APPENDIX A DISTRIBUTION OF A SCALAR PRODUCT

Let  $\mathbf{Y}$  be a random vector uniformly distributed on the unit hypersphere in  $\mathbb{R}^d$  ( $\|\mathbf{Y}\| = 1$ ), and  $\mathbf{m}$  a fixed vector. This section studies the distribution of  $S = \mathbf{Y}^\top \mathbf{m}$ . To generate  $\mathbf{Y}$ , we can first generate a multivariate Gaussian vector  $\mathbf{G} = (G_1, \dots, G_d)^\top \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_d)$ , where  $\mathbf{I}_d$  is  $d \times d$  identity matrix, and set  $\mathbf{Y} = \mathbf{G}/\|\mathbf{G}\|$ . This means that  $G_i$  are i.i.d. Gaussian distributed:  $G_i \sim \mathcal{N}(0, 1)$ . Without loss of generality, using symmetry of the Euclidean norm, assume that  $\mathbf{m} = (\|\mathbf{m}\|, 0, \dots, 0)$ . This simplifies into

$$S = \mathbf{Y}^\top \mathbf{m} = \|\mathbf{m}\| \frac{G_1}{\sqrt{\sum_{i=1}^d G_i^2}}. \quad (18)$$

Obviously,  $-\|\mathbf{m}\| \leq S \leq \|\mathbf{m}\|$  so that its cdf  $F_S(s)$  equals 0 if  $s \leq -\|\mathbf{m}\|$ , and 1 if  $s \geq \|\mathbf{m}\|$ .

For all  $s \in [0, \|\mathbf{m}\|]$ ,  $\mathbb{P}(S \geq s)$  is the probability that vector  $\mathbf{G}$  belongs to the convex cone pointing in the direction  $\mathbf{m}$  and with angle  $\cos^{-1}(s/\|\mathbf{m}\|)$ . By a symmetry argument (replacing  $\mathbf{m}$  by  $-\mathbf{m}$ ),  $\mathbb{P}(S \leq -s) = \mathbb{P}(S \geq s)$ , giving

$$F_S(-s) = 1 - F_S(s), \quad \forall 0 \leq s \leq \|\mathbf{m}\|. \quad (19)$$

It also implies that

$$\mathbb{P}(S^2 \geq s^2) = \mathbb{P}(S \geq s) + \mathbb{P}(S \leq -s) = 2(1 - F_S(s)). \quad (20)$$

Going back to  $\mathbf{G}$ , we can write

$$\begin{aligned} \mathbb{P}(S^2 \geq s^2) &= \mathbb{P}\left(\frac{G_1^2}{\sum_{i=1}^d G_i^2} \geq \tau^2\right) \\ &= \mathbb{P}\left(\frac{G_1^2}{\sum_{i=2}^d G_i^2} \geq \frac{\tau^2}{1 - \tau^2}\right), \end{aligned} \quad (21)$$

with  $\tau = s/\|\mathbf{m}\|$ . By definition, the random variable  $U = (d-1)G_1^2 / \sum_{i=2}^d G_i^2$  has an F-distribution  $F(1, d-1)$ . It follows that

$$\mathbb{P}(S^2 \geq \tau^2 \|\mathbf{m}\|^2) = 1 - I_{\tau^2}(1/2, (d-1)/2), \quad (22)$$

where  $I_x(a, b)$  is the regularized incomplete beta function. In the end,  $\forall s \in [0, \|\mathbf{m}\|]$ , we have

$$F_S(s) = \left(1 + I_{\frac{s^2}{\|\mathbf{m}\|^2}}\left(\frac{1}{2}, \frac{d-1}{2}\right)\right) / 2. \quad (23)$$

By taking the derivative of this expression and accounting for symmetry, the pdf is found to be:

$$f_S(s) = \frac{(1 - s^2/\|\mathbf{m}\|^2)^{\frac{d-3}{2}}}{\|\mathbf{m}\| B(1/2, (d-1)/2)}, \quad \forall s, -\|\mathbf{m}\| \leq s \leq \|\mathbf{m}\|$$

where  $B(a, b)$  is the beta function. This implies that  $\mathbb{E}[S] = 0$  and

$$\begin{aligned} \mathbb{V}[S] &= \frac{\|\mathbf{m}\|^2}{B\left(\frac{1}{2}, \frac{d-1}{2}\right)} \int_0^1 a^2 (1 - a^2)^{\frac{d-3}{2}} da \\ &= \|\mathbf{m}\|^2 \frac{B\left(\frac{1}{2}, \frac{d-3}{2}\right)}{B\left(\frac{1}{2}, \frac{d-1}{2}\right)} = \frac{\|\mathbf{m}\|^2}{d}. \end{aligned}$$

When  $d \rightarrow \infty$ , using the expansion [28, Eq. (26)] of the regularized incomplete beta function in (23) yields

$$F_S(s) \approx \Phi\left(\sqrt{\frac{d-1}{\|\mathbf{m}\|^2}} \frac{2s}{1 + \sqrt{1 - s^2/\|\mathbf{m}\|^2}}\right), \quad (24)$$

which is approximately  $\Phi(\sqrt{d/\|\mathbf{m}\|^2}s)$  for small  $s$ , i.e., the cdf of a centered Gaussian r.v. with variance  $\|\mathbf{m}\|^2/d$ .

## APPENDIX B EXPECTED VALUE OF THE MEMORY UNIT

We assume that the vectors  $\mathbf{x}_i$  are not related to each other so that, for  $n < d$ ,  $\mathbf{X}$  has  $n$  linearly independent columns. Then  $\mathbf{X}^+ = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top$  and  $\|\mathbf{m}^*\|^2 = \mathbf{1}_n^\top (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{1}_n$ . The  $n \times n$  Gram matrix  $\mathbf{X}^\top \mathbf{X}$  is real, symmetric, and positive semi-definite, and so it has an eigendecomposition  $\mathbf{U} \mathbf{\Lambda} \mathbf{U}^\top$ , where  $\mathbf{\Lambda}$  is a diagonal matrix with non-negative coefficients  $\{\lambda_i\}_{i=1}^n$  and  $\mathbf{U} \mathbf{U}^\top = \mathbf{U}^\top \mathbf{U} = \mathbf{I}_n$ . Moreover,  $\text{tr}(\mathbf{X}^\top \mathbf{X}) = \sum_{i=1}^n \lambda_i = n$  because  $\mathbf{x}_i^\top \mathbf{x}_i = 1$ , for all  $i$ .

The first construction  $\mathbf{m} = \mathbf{X} \mathbf{1}_n$  has a square norm  $\|\mathbf{m}\|^2 = \mathbf{1}_n^\top \mathbf{X}^\top \mathbf{X} \mathbf{1}_n = \mathbf{1}_n^\top \mathbf{U} \mathbf{\Lambda} \mathbf{U}^\top \mathbf{1}_n$ . The second construction  $\mathbf{m}^* = \mathbf{X} (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{1}_n$  has a norm  $\|\mathbf{m}^*\|^2 = \mathbf{1}_n^\top (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{1}_n = \mathbf{1}_n^\top \mathbf{U} \mathbf{\Lambda}^{-1} \mathbf{U}^\top \mathbf{1}_n$ . It is difficult to say anything more for a given  $\mathbf{X}$ . However, if we consider  $\mathbf{X}^\top \mathbf{X}$  as a random matrix, then  $\mathbb{E}[(\mathbf{X}^\top \mathbf{X})^{-1}] - (\mathbb{E}[\mathbf{X}^\top \mathbf{X}])^{-1} = \mathbb{E}[(\mathbf{X}^\top \mathbf{X})^{-1}] - \mathbf{I}_n$  is positive semi-definite [29]. This shows that  $\mathbb{E}[\|\mathbf{M}^*\|^2] \geq \mathbb{E}[\|\mathbf{M}\|^2] = n$ . This means that the second construction increases the variance of the score under  $\mathcal{H}_0$ .

To make further progress, we resort to the asymptotic theory of random matrices, and especially to the Marcenko-Pastur distribution. Suppose that both  $d$  and  $n$  tend to infinity while remaining proportional:  $n = cd$  with  $c < 1$ . The matrix  $\mathbf{X}^\top \mathbf{X}$  can be thought of as an empirical covariance matrix of the  $d$  vectors which are the rows of  $\mathbf{X}$ . These vectors are i.i.d. with bounded support components. Therefore, the marginal distribution of the  $n$  eigenvalues  $\{\lambda_i\}_{i=1}^n$  of  $\mathbf{X}^\top \mathbf{X}$  asymptotically follows the Marcenko-Pastur distribution: for all  $\lambda$  such that  $(1 - \sqrt{c})^2 \leq \lambda \leq (1 + \sqrt{c})^2$ ,

$$f_{\text{MP}}(\lambda) = \frac{\sqrt{(\lambda - (1 - \sqrt{c})^2)((1 + \sqrt{c})^2 - \lambda)}}{2c\pi\lambda}. \quad (25)$$

Moreover, for any function  $\psi$  bounded over the interval  $[(1 - \sqrt{c})^2, (1 + \sqrt{c})^2]$ :

$$\frac{1}{n} \sum_{i=1}^n \psi(\lambda_i) - \int \psi(\lambda) f_{\text{MP}}(\lambda) d\lambda \rightarrow 0. \quad (26)$$

Using the eigendecomposition,  $\mathbf{X}^\top \mathbf{X} = \mathbf{U} \mathbf{\Lambda} \mathbf{U}^\top$ , we get  $(\mathbf{X}^\top \mathbf{X})^{-1} = \mathbf{U} \mathbf{\Lambda}^{-1} \mathbf{U}^\top$ , where the columns of  $\mathbf{U}$  are the random eigenvectors. Then, asymptotically as  $d \rightarrow \infty$ :

$$n^{-1} \mathbb{E}[\mathbf{1}_n^\top (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{1}_n] - \int \lambda^{-1} f_{\text{MP}}(\lambda) d\lambda \rightarrow 0. \quad (27)$$

This shows that  $\mathbb{E}[\|\mathbf{M}^*\|^2]/n$  converges to  $\frac{1}{1-c}$  asymptotically (the above integral is the Stieljes transform of the Marcenko-Pastur distribution evaluated at  $z = 0$ ), whereas  $\mathbb{E}[\|\mathbf{M}\|^2]/n$  converges to  $\mathbb{E}[\lambda] = 1$ . In expectation,  $\mathbf{M}^*$  has a higher variance, but only by a factor of  $1/(1-c)$  which remains acceptable if  $c = n/d$  is small.

## APPENDIX C

## Y UNIFORMLY DRAWN OVER A SPHERICAL CAP

Assume that  $\mathbf{Y}$  is uniformly distributed over the spherical cap  $\mathcal{C}_{\mathbf{u}, \gamma}$ , which is the intersection of the unit hypersphere and the single hypercone of axis  $\mathbf{u}$ ,  $\|\mathbf{u}\| = 1$  and angle  $\gamma$ . In other words,  $\|\mathbf{Y}\| = 1$  and  $S' = \mathbf{Y}^\top \mathbf{u} > \cos(\gamma)$ . Denote  $\eta = \cos(\gamma)$  and  $\bar{\eta} = \text{sign}(\eta)$ . The probability distribution function of  $S'$  is

$$f_{S'}(s') = \frac{f_S(s')}{1 - F_S(s')} \mathbb{1}_{[s' > \eta]}(s'). \quad (28)$$

This stems into

$$\mathbb{E}[S'] = \frac{2(1 - \eta^2)^{\frac{d-1}{2}}}{(d-1)B(\frac{1}{2}, \frac{d-1}{2}) (1 - \bar{\eta}I_{\eta^2}(\frac{1}{2}, \frac{d-1}{2}))}. \quad (29)$$

Note that:

- $\eta = -1$ :  $\mathbb{E}[S'] = 0$ . The cap is the full hypersphere.
- $\eta \rightarrow 1$ :  $\mathbb{E}[S'] \rightarrow 1$ , thanks to De l'Hospital's rule. The cap reduces to  $\{\mathbf{u}\}$ .

In the same way:

$$\mathbb{E}[S'^2] = \frac{1}{d} \frac{1 - \bar{\eta}I_{\eta^2}(\frac{3}{2}, \frac{d-1}{2})}{1 - \bar{\eta}I_{\eta^2}(\frac{1}{2}, \frac{d-1}{2})}, \quad (30)$$

from which we can deduce  $\mathbb{V}(S')$  with the König-Huygens formula. Note that:

- $\eta = -1$ :  $\mathbb{V}[S'] = 1/d$ .
- $\eta \rightarrow 1$ :  $\mathbb{V}[S'] \rightarrow 0$ , thanks to De l'Hospital's rule.

From now on, we define

$$\mu_\kappa(\eta, d) = \mathbb{E}[(S')^\kappa], \quad (31)$$

with  $S' = \mathbf{Y}^\top \mathbf{u}$  and  $\mathbf{Y} \sim \mathcal{U}_{\mathcal{C}_{\mathbf{u}, \gamma}}$ .

## APPENDIX D

## MODELING K-MEANS FOR THE SUM CONSTRUCTION

We assume that  $k$ -means has packed together in a memory unit independent vectors uniformly distributed over the spherical cap:  $\mathbf{x}_i \sim \mathcal{U}_{\mathcal{C}_{\mathbf{u}, \gamma}}$ . Vector  $\mathbf{x}_i$  can be modeled as  $\mathbf{x}_i = S'_i \mathbf{u} + \mathbf{N}_i$ , where  $\{S'_i\}_{i=1}^n$  are i.i.d. according to the pdf described in Appendix C,  $\mathbf{N}_i^\top \mathbf{u} = 0$  and  $\|\mathbf{N}_i\|^2 = 1 - (S'_i)^2$ . Their memory vector is the sum  $\mathbf{m} = \sum_{i=1}^n \mathbf{x}_i$ .

A. Hypothesis  $\mathcal{H}_0$ 

$\mathbf{Y}$  is independent from  $\mathbf{m}$ . It follows that:

$$\mathbb{E}[\mathbf{Y}^\top \mathbf{m}] = 0, \quad \mathbb{V}[\mathbf{Y}^\top \mathbf{m}] = \mathbb{E}[\|\mathbf{m}\|^2]/d \quad (32)$$

with

$$\mathbb{E}[\|\mathbf{m}\|^2] = \mathbb{E}\left[\sum_{i=1}^n \|\mathbf{x}_i\|^2 + 2 \sum_{i < j} S'_i S'_j + \mathbf{N}_i^\top \mathbf{N}_j\right] \quad (33)$$

$$= n + n(n-1)\mu_1(\eta, d)^2, \quad (34)$$

where  $\mu_1(\eta, d)$  is given in (29).

To sum up: while  $\mathbb{E}[\mathbf{Y}^\top \mathbf{m}] = 0$ , the variance is increasing with  $\eta$ .

- $\eta = -1$ : It equals  $n/d$  as already shown by (4).
- $\eta \rightarrow 1$ : It converges to  $n^2/d$ .

B. Hypothesis  $\mathcal{H}_1$ 

We now single out the role of the matching vector  $\mathbf{X}_1$ :  $\mathbf{m} = \mathbf{x}_1 + \mathbf{m}'$ . The query is modeled as  $\mathbf{Y} = \alpha \mathbf{x}_1 + \beta \mathbf{Z}$  with  $\|\mathbf{Z}\| = 1$  and  $\mathbf{Z}^\top \mathbf{x}_1 = 0$ . With the same notation as above:

$$\mathbf{Y}^\top \mathbf{m} = \alpha + \alpha \sum_{i=2}^n S_i S_i + \alpha \mathbf{N}_1^\top \mathbf{m}' + \beta \mathbf{Z}^\top \mathbf{m}'. \quad (35)$$

The expectation easily comes as

$$\mathbb{E}[\mathbf{Y}^\top \mathbf{m}] = \alpha(1 + (n-1)\mu_1(\eta, d)^2). \quad (36)$$

The variance of the second summand is given by the law of total variance:

$$\mathbb{V}[\alpha \sum_{i=2}^n S_i S_i] = \alpha^2(n-1)(\mu_2(\eta, d)^2 - \mu_1(\eta, d)^4). \quad (37)$$

The variance of the third term is

$$\mathbb{V}[\alpha \sum_{i=2}^n \mathbf{N}_i^\top \mathbf{N}_i] = \frac{\alpha^2(n-1)}{d-1}(1 - \mu_2(\eta, d)^2). \quad (38)$$

The variance of the last summand is more complex to analyze. We need to decompose  $\mathbf{Z}$  into its projection on  $\mathbf{u}$  and on the complementary space.

$$\begin{aligned} \mathbb{V}[\beta(\mathbf{Z}^\top \mathbf{u})\mathbf{u}^\top \mathbf{m}'] &= (1 - \alpha^2) \frac{n-1}{d-1} (1 - \mu_2(\eta, d)) \mu_2(\eta, d) \\ \mathbb{V}[\beta \mathbf{Z}_\perp^\top \mathbf{m}'] &= (1 - \alpha^2) \frac{n-1}{d-1} (1 - \mu_2(\eta, d)) \\ &\quad \times (1 + (n-2)\mu_1(\eta, d)^2) \end{aligned}$$

To sum up:  $\mathbb{E}[\mathbf{Y}^\top \mathbf{m}]$  increases while  $\mathbb{V}[\mathbf{Y}^\top \mathbf{m}]$  decreases with  $\eta$ .

- $\eta = -1$ :  $\mathbb{E}[\mathbf{Y}^\top \mathbf{m}] = \alpha$  while  $\mathbb{V}[\mathbf{Y}^\top \mathbf{m}] = (n-1)/d$  as already shown by (5).
- $\eta \rightarrow 1$ :  $\mathbb{E}[\mathbf{Y}^\top \mathbf{m}] \rightarrow n\alpha$  whereas  $\mathbb{V}[\mathbf{Y}^\top \mathbf{m}] \rightarrow 0$ .

In the end, under the Gaussian assumption, the Kullback-Leibler distance between both distributions increases which proves that identifying the positive memory units becomes easier as  $\eta$  increases (see Fig. 17).

## APPENDIX E

## MODELING K-MEANS FOR THE PINV CONSTRUCTION

A. Hypothesis  $\mathcal{H}_0$ 

We make the same assumption as in Appendix D. We write  $\mathbf{X}$  as  $\mathbf{X} = \mathbf{u} \mathbf{S}'^\top + \mathbf{N}$  with  $\mathbf{S}'$  a  $n \times 1$  vector storing the correlations  $\mathbf{u}^\top \mathbf{x}_i$ ,  $1 \leq i \leq n$  and  $\mathbf{N}$  a  $d \times n$  matrix whose columns are random vectors orthogonal to  $\mathbf{u}$  and of norm  $\sqrt{1 - S_i'^2}$ . The memory unit is now given by (9):  $\mathbf{m}^* = \mathbf{X}(\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{1}_n$ . Eq. (32) holds but with a new expression for the norm of  $\mathbf{m}^*$ :

$$\mathbb{E}[\|\mathbf{m}^*\|^2] = \mathbb{E}[\mathbf{1}_n^\top (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{1}_n] \quad (39)$$

$$\geq \mathbf{1}_n^\top (\mathbb{E}[\mathbf{X}^\top \mathbf{X}])^{-1} \mathbf{1}_n. \quad (40)$$

We write matrix  $\mathbf{X}^\top \mathbf{X} = \mathbf{X}^\top \mathbf{X} = \mathbf{S}' \mathbf{S}'^\top + \mathbf{N} \mathbf{N}^\top$  s.t.:

$$\mathbb{E}[\mathbf{X}^\top \mathbf{X}] = (1 - \mathbb{E}[S'^2]) \mathbf{I}_n + \mathbb{E}[S'^2] \mathbf{1}_n \mathbf{1}_n^\top, \quad (41)$$

whose inverse is given by the Sherman-Morrison formula:

$$\mathbb{E}[\mathbf{X}^\top \mathbf{X}]^{-1} = \frac{1}{1 - \mathbb{E}[S'^2]} \left( \mathbf{I}_n + \frac{\mathbb{E}[S'^2]}{1 + (n-1)\mathbb{E}[S'^2]} \mathbf{1}_n \mathbf{1}_n^\top \right),$$

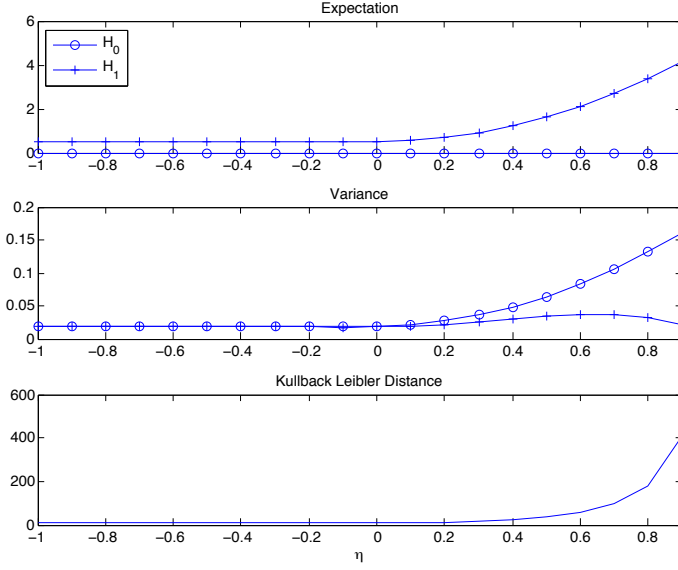


Fig. 17. Expectations and Variances under both hypothesis and Kullback Leibler distance as functions of  $\eta$  for the sum construction.  $\alpha = 0.5$ ,  $d = 512$ .

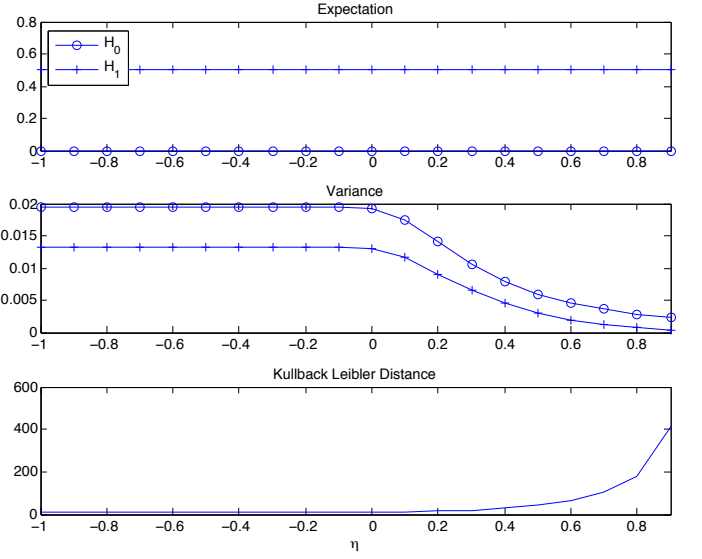


Fig. 18. Expectations and Variances under both hypothesis and Kullback Leibler distance as functions of  $\eta$  for the pinv construction.  $\alpha = 0.5$ ,  $d = 512$ .

leading to

$$\mathbb{E}[\|\mathbf{m}^*\|^2] \geq \frac{n}{1 + (n-1)\mathbb{E}[S']^2} \quad (42)$$

To sum up: While  $\mathbb{E}[\mathbf{Y}^\top \mathbf{m}^*]$  remains constant, the lower bound of the variance is decreasing with  $\eta$ .

- $\eta = -1$ : The lower bound equals  $\frac{n}{d}$  which is tight w.r.t. to the previous result in Sect. B (i.e.  $\frac{n}{d}(1 + \frac{n}{(d-n)})$ ) for  $n$  much smaller than  $d$ .
- $\eta \rightarrow 1$ : The lower bounds converges to  $1/d$  and is also tight since  $\mathbf{m}^* \rightarrow \mathbf{x}_1$  as the vectors are converging to  $\mathbf{x}_1$ .

### B. Hypothesis $\mathcal{H}_1$

We single out the matching vector  $\mathbf{x}_1$  by writing  $\mathbf{X} = (\mathbf{x}_1, \bar{\mathbf{X}})$  with  $\bar{\mathbf{X}} = \mathbf{u}\mathbf{S}'^\top + \mathbf{N}$ ,  $\mathbf{S}'$  being now a  $(n-1) \times 1$  vector storing the correlations  $\mathbf{u}^\top \mathbf{x}_i$ ,  $2 \leq i \leq n$  and  $\mathbf{N}$  a  $d \times (n-1)$  matrix whose columns are random vectors orthogonal to  $\mathbf{u}$  and of norm  $\sqrt{1 - S_i'^2}$ . This makes

$$\mathbf{X}^\top \mathbf{X} = \begin{pmatrix} 1 & \mathbf{x}_1^\top \bar{\mathbf{X}} \\ \bar{\mathbf{X}}^\top \mathbf{x}_1 & \bar{\mathbf{X}}^\top \bar{\mathbf{X}} \end{pmatrix}, \quad (43)$$

whose inverse is

$$(\mathbf{X}^\top \mathbf{X})^{-1} = \begin{pmatrix} 1 + \mathbf{x}_1^\top \bar{\mathbf{X}} \mathbf{D} \bar{\mathbf{X}}^\top \mathbf{x}_1 & -\mathbf{x}_1^\top \bar{\mathbf{X}} \mathbf{D} \\ -\mathbf{D} \bar{\mathbf{X}}^\top \mathbf{x}_1 & \mathbf{D} \end{pmatrix}. \quad (44)$$

with  $\mathbf{D} = (\bar{\mathbf{X}}^\top (\mathbf{I} - \mathbf{X}_1 \mathbf{X}_1^\top) \bar{\mathbf{X}})^{-1}$ . This makes the following memory vector:

$$\mathbf{m}^* = \mathbf{x}_1 + \mathbf{m}_\perp^* \quad (45)$$

$$\mathbf{m}_\perp^* = (\mathbf{I} - \mathbf{x}_1 \mathbf{x}_1^\top) \bar{\mathbf{X}} \mathbf{D} (\mathbf{1}_{n-1} - \bar{\mathbf{X}} \mathbf{x}_1) \quad (46)$$

Note that  $\|\mathbf{m}^*\|^2 = 1 + \|\mathbf{m}_\perp^*\|^2$  because  $\mathbf{x}_1^\top \mathbf{m}_\perp^* = 0$ .

The query vector being defined as in Sect. D-B, its correlation with the memory vector is

$$\mathbf{Y}^\top \mathbf{m}^* = \alpha + \sqrt{1 - \alpha^2} \mathbf{Z}^\top \mathbf{m}_\perp^*, \quad (47)$$

whose expectation and variance are given by

$$\mathbb{E}[\mathbf{Y}^\top \mathbf{m}^*] = \alpha, \quad \mathbb{V}[\mathbf{Y}^\top \mathbf{m}^*] = (1 - \alpha^2) \frac{\mathbb{E}[\|\mathbf{m}_\perp^*\|^2]}{d-1} \quad (48)$$

with

$$\mathbb{E}[\|\mathbf{m}_\perp^*\|^2] = \mathbb{E}[\|\mathbf{m}^*\|^2] - 1 \quad (49)$$

$$\geq \frac{(n-1)(1 - \mathbb{E}[S']^2)}{1 + (n-1)\mathbb{E}[S']^2}. \quad (50)$$

To sum up: While  $\mathbb{E}[\mathbf{Y}^\top \mathbf{m}^*]$  remains constant, the lower bound of the variance is decreasing with  $\eta$ .

- $\eta = -1$ : The lower bound equals  $(1 - \alpha^2) \frac{n-1}{d-1}$  which is tight w.r.t. to the previous result in Sect. B (i.e.  $(1 - \alpha^2) \frac{n}{d-n}$ ) for  $n$  much smaller than  $d$ .
- $\eta \rightarrow 1$ : The lower bounds converges to 0 and is also tight since  $\mathbf{m}^* \rightarrow \mathbf{x}_1$  as the vectors are converging to  $\mathbf{x}_1$ .

In the end, under the Gaussian assumption, the Kullback-Leibler distance between both distributions increases which proves that identifying the positive memory units becomes easier as  $\eta$  increases (see Fig. 18).